AMENDMENTS TO THE SPECIFICATION

In paragraph [0011] of the originally filed application, please amend as reflected in the following, marked-up version of the paragraph:

[0011] The personal computer 20 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 49. The remote computer 49 may be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the personal computer 20, although only a memory storage device 50 has been illustrated in FIG. 1. The logical connections depicted in FIG. 1 include a local area network (LAN) 51 and a wide area network (WAN)~~—52~~. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

In paragraph [0012] of the originally filed application, please amend as reflected in the following, marked-up version of the paragraph:

[0012] When used in a LAN networking environment, the personal computer 20 is connected to the local network 51 through a network interface or adapter 53. When used in a WAN networking environment, the personal computer 20 typically includes a modem 54 or other means for establishing communications over the WAN~~—52~~. The modem 54, which may be internal or external, is connected to the system bus 23 via the serial port interface 46. In a networked environment, program modules depicted relative to the personal computer 20, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

In paragraph [0014] of the originally filed application, please amend as reflected in the following, marked-up version of the paragraph:

[0014] The present invention is directed to a new approach to accessing objects in a database, such a directory service database, by expressing the relationships among attributes of the objects in the database to allow the objects in the database to be located based on different types of relationships among the object attributes. This concept of viewing different types of relationships with the same set of data is termed "polyarchy." The polyarchy concept of the invention enables clients to query and access the database objects in ways that are much more powerful than conventional database queries, such as LDAP filters and even the Structured Query Language (SQL) queries. As a direct result of the polyarchical arrangement of the database, a new form of database query expression can be used to identify a path through the database objects for locating the desired data. Such an expression of a path is hereinafter referred to as a "location path expression."

In paragraph [0022] of the originally filed application, please amend as reflected in the following, marked-up version of the paragraph:

[0022] In accordance with a feature of the invention, once the relationships between attributes of the classes for the database objects are defined, they can be used for locating objects and their attributes in the database by providing a path that uses the relationships between the attributes to lead to the desired data. This new approach to expressing a query for database access is significantly different from the conventional model, such as LDAP, for accessing data in a database. As will become clear from the discussion below, this new enables the use of rich, multi-attribute, queries on a directory service or similar databases containing different types of objects with linked attributes. Rich queries can be formed that are difficult to do using conventional directory service query approaches. For instance, as will be explained in greater detail below, the query expressions in may be along the line of "list those employees reporting to Steve Smith who are allowed access to resource B."

In paragraph [0023] of the originally filed application, please amend as reflected in the following, marked-up version of the paragraph:

[0023] Referring to FIG. 5, in a preferred embodiment, the path expressions for locating desired database data have a format that is generally similar to the syntax of the XML PATH Language (XPath). XPath is defined in XML Path Language Version 1.0 by World Wide Web Consortium (W3C), which is hereby incorporated by reference in its entirety. XPath is a component of the Extensible Stylesheet Language (XSL) that is used to identify tagged XML elements and attributes in an XML document. It is also used to calculate numbers and manipulate strings. As shown in FIG. 5, the data path expression starts with a root node of the path, represented by the ~~backslash~~ forward slash ("/") symbol. Following the root node symbol 136 is "ViewName" parameter 138. The ViewName is followed by a path element 140, which is separated from the ViewName by a forward slash. The path element 140 may be followed by another path element 146, and so on, with the string of path elements separated by forward slashes. Each of the path element forms a node in the search path and may be the attribute value of a database object, or special symbols, such as wildcard symbols or reverse search, etc., to indicate a search strategy.

In paragraph [0029] of the originally filed application, please amend as reflected in the following, marked-up version of the paragraph:

[0029] Referring to FIG. 6, in a preferred embodiment, the client is provided with Application Programming Interface (API) functions 150 that the application 152 on the client machine can call to form the data path expressions and queries using such expressions for accessing the directory service. The queries using the data path expressions are put in a request message 160 that is sent via SOAP over HTTP or other transport protocol to the Web service 92 for directory access. The Web service 92 includes a module 162 for converting the queries based on the data path expressions into LDAP queries 168 for carrying out the requested directory data access. The LDAP engine 166 of the Web service then sends the LDAP queries 168 to the database server 72 to retrieve the object data. It will be appreciated that although the data path expression may appear to be simple, the corresponding LDAP queries may actually be quite complicated and may require recursive or iterative execution of some queries to locate and retrieve all data specified by one data path expression. It can be seen that a significant advantage of this arrangement is that the client does not need to worry about traversing the data links defined by the path itself, because the database engine does the complex queries on behalf of the client. Having this architecture allows several benefits. First, it reduces the traffic between the client and the server, since the server will return the result 170 that has already been processed. Second, it enables administrative control on the server, which could prevent unauthorized access or restrict the access of an abusive user of the system. Moreover, the server could optimize the query, not being limited to the LDAP filters. The server could internally implement a query processor optimized for carrying out the database search based on location path expressions.

In paragraph [0030] of the originally filed application, please amend as reflected in the following, marked-up version of the paragraph:

[0030] To illustrate how the location path expressions are used in searching for the desired data, two examples are provided here. The Manager-DirectReports relationship can be modeled as:

Source:Class:Person Attribute:DirectReports

Target:Class:Person Attribute:~~Manger~~ Manager

By having this information, a parser of the database engine can easily traverse up and down a path, such as /OrgChart/John/Jane/Alice. In this example, the view name "OrgChart" has been defined to indicate that the view is based on the Manager-DirectReports relationship. Given this location path expression, the database engine first looks at the definition of OrgChart and learns that the relationship to be used for viewing is Manager-DirectReports, which is defined between Person objects. Thus, the first element in the location path after the view name should be a person that is named "John." The database engine locates the Person object for John, and looks at the DirectReports attribute of John to find the Person object for Jane. Once the Jane object is found, it locates the Person object for Alice by looking at the DirectReports attribute of Jane. In another example, the location path expression is "/OrgChart/Alice/..", where ".." denotes going up to a parent node. The search process is similar to that of the previous example, except that in this example, there is a reversal in the search direction due to the ".." element. Once the database engine finds the Person object for Alice, it tries to find Alice's manager by looking at the Manager attribute of Alice. As a result, it will find Jane.